US HOUSE PRICE PREDICTION

MACHINE LEARNING MODEL

dimpybangoriya@gmail.com

ABOUT THE PROJECT

This project involves building a machine learning model to predict house prices in the United States based on various factors such as location, square footage, number of bedrooms and bathrooms, and other property features. Using Python, data was preprocessed, exploratory data analysis (EDA) was performed, and regression model was trained to achieve accurate predictions. Key libraries such as Pandas, Numpy, Scikit-Learn, and Matplotlib were used for data manipulation, modeling, and visualization.

The project aims to help real estate professionals and buyers make data-driven decisions by providing reliable price estimations based on historical trends and market patterns.

1. STUDYING DATASET

[2]:
```
import numpy as np
import pandas as pd
import seaborn as sns
```

[3]:
```
import matplotlib.pyplot as plt
```

[10]:
```
df=pd.read_csv('Class Data 1 Project Housing Price Prediction W9 L2 23_1_2024.csv')
```

[12]:
```
df.head()
```

[12]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Views Suite 079\nLake Kathleen, CA...
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabeth Stravenue\nDanieltown, WI 06482...
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFPO AP 44820
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nFPO AE 09386

[14]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   Avg. Area Income             5000 non-null   float64
 1   Avg. Area House Age          5000 non-null   float64
 2   Avg. Area Number of Rooms    5000 non-null   float64
 3   Avg. Area Number of Bedrooms 5000 non-null   float64
 4   Area Population              5000 non-null   float64
 5   Price                        5000 non-null   float64
 6   Address                      5000 non-null   object 
dtypes: float64(6), object(1)
memory usage: 273.6+ KB
```

[ ]:

2. DIVIDING DATA IN INPUT & OUTPUT

```
[ ]: #creating machine learning model

[ ]:

[ ]: # STEP 1:  DIVIDE DATA INTO 2 PARTS INPUT y AND OUTPUT X - 2 PARTS

[ ]:

[ ]:    # OUTPUT

•[16]:  y=df['Price']

[ ]:

[ ]:    # INPUT

•[29]:  X=df[['Avg. Area Income','Avg. Area House Age','Avg. Area Number of Rooms','Avg. Area Number of Bedrooms','Area Population']]

[31]: X
```

[31]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population
0	79545.458574	5.682861	7.009188	4.09	23086.800503
1	79248.642455	6.002900	6.730821	3.09	40173.072174
2	61287.067179	5.865890	8.512727	5.13	36882.159400
3	63345.240046	7.188236	5.586729	3.26	34310.242831
4	59982.197226	5.040555	7.839388	4.23	26354.109472
...	...	...	...	...	...
4995	60567.944140	7.830362	6.137356	3.46	22837.361035
4996	78491.275435	6.999135	6.576763	4.02	25616.115489
4997	63390.686886	7.250591	4.805081	2.13	33266.145490
4998	68001.331235	5.534388	7.130144	5.44	42625.620156
4999	65510.581804	5.992305	6.792336	4.07	46501.283803

5000 rows × 5 columns

3. DIVIDING INPUT & OUTPUT INTO TEST AND TRAIN DATA

```
[ ]:
```

```
[15]: # STEP 2 Divide input - X and output - y - into 4 parts train/test data X_train,X_test,y_train,y_test
```

```
[ ]:
```

```
[16]: from sklearn.model_selection import train_test_split
```

```
[17]: X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)
```

```
[18]: X_train
```

[18]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population
4227	66547.016454	5.846095	6.847298	4.13	27850.822901
4676	53722.008599	6.401391	7.787764	3.30	47649.224665
800	64838.492899	6.437157	8.699544	4.02	32921.010068
3671	67097.092120	6.086754	7.211963	3.05	27191.506877
4193	75245.465436	8.167820	7.420100	3.42	37410.669928
...	...	...	...	...	...
4426	76223.561256	6.371627	5.342217	2.42	30165.337445
466	56685.014442	6.958045	7.502115	3.38	43322.166854
3092	66195.337714	6.507971	6.611861	3.14	37288.923574
3772	58694.515017	7.394768	9.269453	4.32	49960.977236
860	61162.580254	5.896316	7.880521	6.04	36033.701431

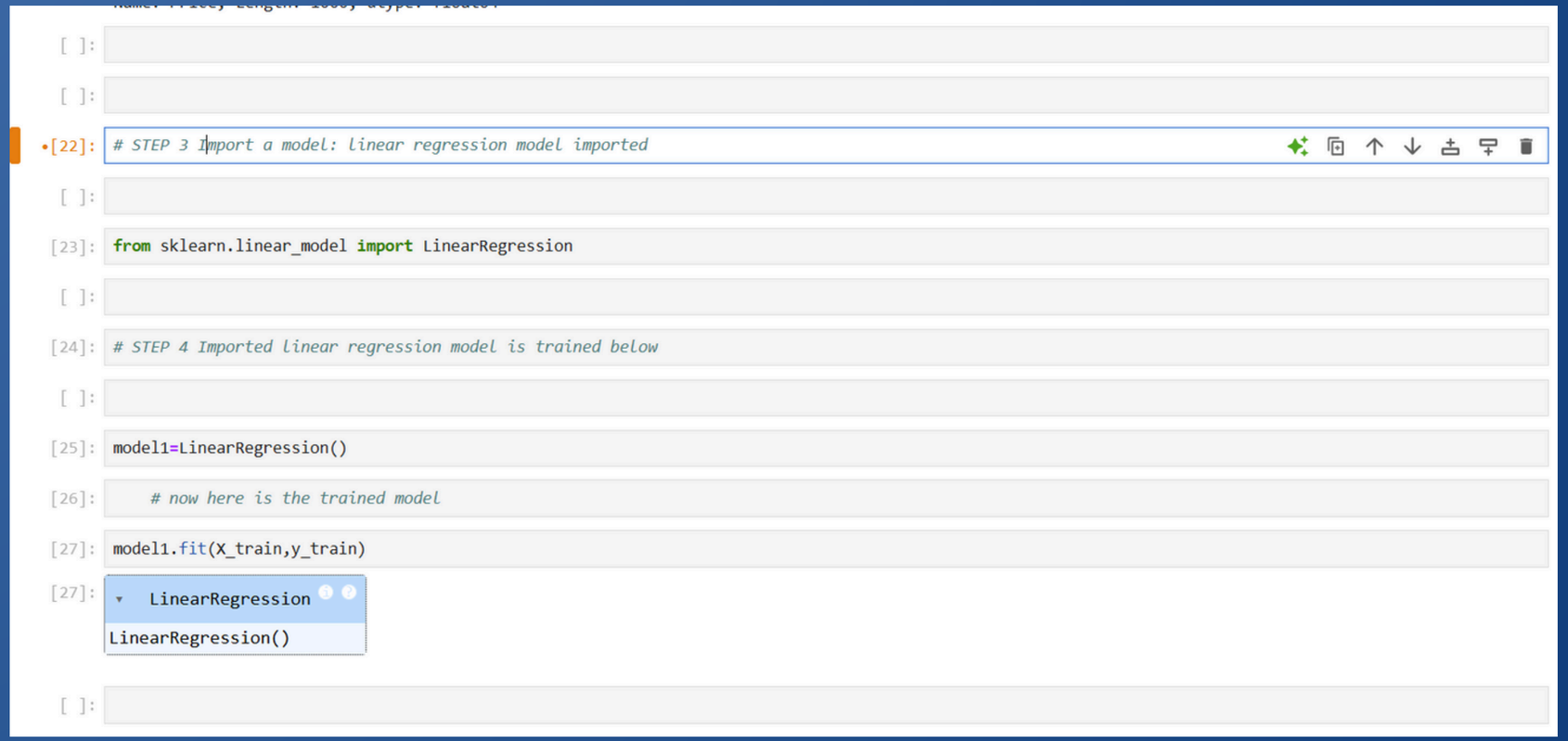
4000 rows × 5 columns

```
[19]: X_test
```

[19]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population
1501	61907.593345	7.017838	6.440256	3.25	43828.947207
2586	57160.202243	6.893260	6.921532	3.13	43467.147035
2653	70190.796445	6.745054	6.662567	2.01	29215.136112
1055	69316.796889	6.300409	7.873576	4.28	24448.211461
705	72991.481649	3.412866	6.494081	2.48	50626.495426

4. IMPORTING A LINEAR REGRESSION MODEL AND TRAINING IT



5. PREDICT GIVING TEST DATA AS INPUT X_TEST

```
[ ]:

[28]: # STEP 5 Predict giving input x_test which is test data

[ ]:

[29]: y_predict=model1.predict(X_test)

[30]: y_predict # y predict has prediction values

[30]: array([1308587.92699753, 1237037.22949429, 1243429.34030687,
       1228900.21360378, 1063320.90710821, 1544058.05034856,
       1094774.70493022,  833284.72339228,  788412.85578723,
       1469714.86615707,  671728.43662066, 1606818.21977937,
       1004166.61331062, 1796798.97595927, 1288566.96221018,
       1087782.93301077, 1423072.37492527, 1078178.68169674,
        802286.03537901,  930761.03695713, 1134829.86477819,
        916398.42023136, 1489972.69335423, 1284580.15538819,
       1582071.35322731, 1132519.15991993, 1089888.39644513,
        974510.51872158,  924057.96820667, 1740759.72092273,
       1286481.5951232 , 1621289.9517161 , 1435264.20161716,
       1234014.77924483, 1485434.57300369, 1718335.00753688,
       1538953.74882847,  777106.64791795, 1765201.52243617,
       1175972.1419982 , 1553707.94323484,  897703.67505177,
       1371049.80326608,  845281.72310358, 1201022.89803884,
       1133285.98450857, 1363128.14557352, 1449814.0876827 ,
       1574363.90467353, 1233577.50265971, 1484464.01606207,
       1295276.5894355 , 1222136.77335286,  990124.41659784,
       1693824.96035767, 1823785.05665098, 1136495.63903357,
       1282164.40305633, 1327292.05443149, 1353355.51909084,
        966265.44345957,  661906.63917319, 1533750.56861   ,
       1002479.7605366 ,  995799.79959536, 1567349.59707244,
       1500813.63482576, 1090078.00056424, 1820964.84741689,
       1479856.16724363,  902785.3021607 , 1494542.21679588,
       1378859.29762368,  962610.88478571,  712800.76347478,
       1565650.37425306, 1149218.97654372,  931311.21938344,
       1600923.95574326,  506875.42639204, 1592924.03164876,
       1292023.54953344,  681260.98813947,  432977.16110002,
```

6. COMPARE PREDICTED VALUES WITH TEST OUTPUT I.E. Y_TEST

```
[ ]:

[28]: # STEP 5 Predict giving input x_test which is test data

[ ]:

[29]: y_predict=model1.predict(X_test)

[30]: y_predict # y predict has prediction values

[30]: array([1308587.92699753, 1237037.22949429, 1243429.34030687,
       1228900.21360378, 1063320.90710821, 1544058.05034856,
       1094774.70493022,  833284.72339228,  788412.85578723,
       1469714.86615707,  671728.43662066, 1606818.21977937,
       1004166.61331062, 1796798.97595927, 1288566.96221018,
       1087782.93301077, 1423072.37492527, 1078178.68169674,
        802286.03537901,  930761.03695713, 1134829.86477819,
        916398.42023136, 1489972.69335423, 1284580.15538819,
       1582071.35322731, 1132519.15991993, 1089888.39644513,
        974510.51872158,  924057.96820667, 1740759.72092273,
       1286481.5951232 , 1621289.9517161 , 1435264.20161716,
       1234014.77924483, 1485434.57300369, 1718335.00753688,
       1538953.74882847,  777106.64791795, 1765201.52243617,
       1175972.1419982 , 1553707.94323484,  897703.67505177,
       1371049.80326608,  845281.72310358, 1201022.89803884,
       1133285.98450857, 1363128.14557352, 1449814.0876827 ,
       1574363.90467353, 1233577.50265971, 1484464.01606207,
       1295276.5894355 , 1222136.77335286,  990124.41659784,
       1693824.96035767, 1823785.05665098, 1136495.63903357,
       1282164.40305633, 1327292.05443149, 1353355.51909084,
        966265.44345957,  661906.63917319, 1533750.56861   ,
       1002479.7605366 ,  995799.79959536, 1567349.59707244,
       1500813.63482576, 1090078.00056424, 1820964.84741689,
       1479856.16724363,  902785.3021607 , 1494542.21679588,
       1378859.29762368,  962610.88478571,  712800.76347478,
       1565650.37425306, 1149218.97654372,  931311.21938344,
       1600923.95574326,  506875.42639204, 1592924.03164876,
       1292023.54953344,  681260.98813947,  432977.16110002,
```

7. PLOTTING ON GRAPH TO CHECK ACCURACY

```
[ ]:

[28]: # STEP 5 Predict giving input x_test which is test data

[ ]:

[29]: y_predict=model1.predict(X_test)

[30]: y_predict # y predict has prediction values

[30]: array([1308587.92699753, 1237037.22949429, 1243429.34030687,
       1228900.21360378, 1063320.90710821, 1544058.05034856,
       1094774.70493022,  833284.72339228,  788412.85578723,
       1469714.86615707,  671728.43662066, 1606818.21977937,
       1004166.61331062, 1796798.97595927, 1288566.96221018,
       1087782.93301077, 1423072.37492527, 1078178.68169674,
        802286.03537901,  930761.03695713, 1134829.86477819,
        916398.42023136, 1489972.69335423, 1284580.15538819,
       1582071.35322731, 1132519.15991993, 1089888.39644513,
        974510.51872158,  924057.96820667, 1740759.72092273,
       1286481.5951232 , 1621289.9517161 , 1435264.20161716,
       1234014.77924483, 1485434.57300369, 1718335.00753688,
       1538953.74882847,  777106.64791795, 1765201.52243617,
       1175972.1419982 , 1553707.94323484,  897703.67505177,
       1371049.80326608,  845281.72310358, 1201022.89803884,
       1133285.98450857, 1363128.14557352, 1449814.0876827 ,
       1574363.90467353, 1233577.50265971, 1484464.01606207,
       1295276.5894355 , 1222136.77335286,  990124.41659784,
       1693824.96035767, 1823785.05665098, 1136495.63903357,
       1282164.40305633, 1327292.05443149, 1353355.51909084,
        966265.44345957,  661906.63917319, 1533750.56861   ,
       1002479.7605366 ,  995799.79959536, 1567349.59707244,
       1500813.63482576, 1090078.00056424, 1820964.84741689,
       1479856.16724363,  902785.3021607 , 1494542.21679588,
       1378859.29762368,  962610.88478571,  712800.76347478,
       1565650.37425306, 1149218.97654372,  931311.21938344,
       1600923.95574326,  506875.42639204, 1592924.03164876,
       1292023.54953344,  681260.98813947,  432977.16110002,
```

8. REALTIME INPUT GIVEN TO MODEL

```
[ ]:

[28]: # STEP 5 Predict giving input x_test which is test data

[ ]:

[29]: y_predict=model1.predict(X_test)

[30]: y_predict # y predict has prediction values

[30]: array([1308587.92699753, 1237037.22949429, 1243429.34030687,
       1228900.21360378, 1063320.90710821, 1544058.05034856,
       1094774.70493022,  833284.72339228,  788412.85578723,
       1469714.86615707,  671728.43662066, 1606818.21977937,
       1004166.61331062, 1796798.97595927, 1288566.96221018,
       1087782.93301077, 1423072.37492527, 1078178.68169674,
        802286.03537901,  930761.03695713, 1134829.86477819,
        916398.42023136, 1489972.69335423, 1284580.15538819,
       1582071.35322731, 1132519.15991993, 1089888.39644513,
        974510.51872158,  924057.96820667, 1740759.72092273,
       1286481.5951232 , 1621289.9517161 , 1435264.20161716,
       1234014.77924483, 1485434.57300369, 1718335.00753688,
       1538953.74882847,  777106.64791795, 1765201.52243617,
       1175972.1419982 , 1553707.94323484,  897703.67505177,
       1371049.80326608,  845281.72310358, 1201022.89803884,
       1133285.98450857, 1363128.14557352, 1449814.0876827 ,
       1574363.90467353, 1233577.50265971, 1484464.01606207,
       1295276.5894355 , 1222136.77335286,  990124.41659784,
       1693824.96035767, 1823785.05665098, 1136495.63903357,
       1282164.40305633, 1327292.05443149, 1353355.51909084,
        966265.44345957,  661906.63917319, 1533750.56861   ,
       1002479.7605366 ,  995799.79959536, 1567349.59707244,
       1500813.63482576, 1090078.00056424, 1820964.84741689,
       1479856.16724363,  902785.3021607 , 1494542.21679588,
       1378859.29762368,  962610.88478571,  712800.76347478,
       1565650.37425306, 1149218.97654372,  931311.21938344,
       1600923.95574326,  506875.42639204, 1592924.03164876,
       1292023.54953344,  681260.98813947,  432977.16110002,
```

CONCLUSION

This project successfully developed a linear regression machine learning model to predict house prices in the United States based on key property features. By leveraging Python and utilizing libraries like Pandas, NumPy, Scikit-Learn, and Matplotlib, the dataset was cleaned, explored, and used to train regression models. its accuracy was checked using metrics such as R-squared and Mean Squared Error (MSE).

The findings demonstrate that factors like location, square footage, and number of bedrooms and bathrooms significantly influence house prices. The model provides valuable insights for real estate professionals, buyers, and investors, enabling data-driven decision-making for pricing strategies and property valuation.

THANK YOU

dimpybangoriya@gmail.com